Artificial Intelligence Assignment one Report

Haile Dereje……………………UGR/2190 /12
Section -3

Submitted to: Mr. Amanuel Negash
Due Date: 29/05/2022

Analysis Report on Graph search algorithms

- Two evaluate this algorithm I had used the number of nodes it touches to get the target node and the average time it takes to find a solution and the solution length as the test variant

- To find the exact path which leads to initial to final the function uses back track method on visited nodes and returns the path if necessary But my point is on the number of nodes it touches before reaching the final or target node for BFS and DFS and for **A*star** and **Dijsktra's**
  I used a special tracker methode named finder which tracks the path based on neighbors distance from the start

Depth First Search

- I made 2 kinds of implementations which are the Iterative one and the recursive one . I prefer to put the analysis result using the recursive one.

- I used both **time** and **timeit** python libraries

- I used stack like implementation on the data structure which holds the adjacent values

The overall **time** analysis seems like this.

no_iteration	**timeit**() (avg time) (100000)	**time**() (avg time)
400	5.12 ms	4.45 ms
400	5.43 ms	4.87 ms

Table . summarizing the time analisis

So on average DFS takes 5.21 microseconds to find distance between two nodes

The Solution length analysis

- Number of iterations 400
- Total paths discover for 400 iterations is 4848
- So average solution length is :
- Average length = 4848/ 400
- Average length = 11
- So on average DFS need to travel 11 nodes to find the distance between two nodes

Summery

- Generally it takes the DFS touching 11 nodes before reaching the final or destination node and average time of 5.21 ms.

Breadth First Search (BFS)

- **BFS** is a traversing algorithm where you should start traversing from a selected node (source or starting node) and traverse the graph layerwise thus exploring the neighbor nodes (nodes which are directly connected to source node). You must then move towards the next-level neighbor nodes.

Code analysis

no_iteration	**timeit**() (avg time) (100000)	**time**() (avg time)
400	5.2 ms	5.45 ms
400	5.3 ms	6.07 ms

Solution length analysis

- Number of iterations i took 400
- -Total paths discover for 400 iterations is 4200
- So average solution length is 4200/ 400 = 10

So on average BFS needs to travel 10 nodes to find the distance between two nodes . Generally My BFS algorithm takes 5.2 microseconds and 10 paths on average to travel between two nodes

Dijsktra's Algorithm

- Dijkstra's algorithm allows us to find the shortest path between any two vertices of a graph.

- It differs from the minimum spanning tree because the shortest distance between two vertices might not include all the vertices of the graph.

Code analysis

- since dijkstra algorithm founds the shortest path from the node to every node at a time I

- I run timit for 100,000 times to get the optimum average
- Total time i found for 100,000 iterations is 22.8 milliseconds
- so on average 20 iterations take 22.8 /100,000 = 2.28 milliseconds to run
- the average time for traveling from one node to another is therefore:
  Average time = 2.28/400 ,Average time = 5.7 microsecond

Solution length analysis

- Number of iterations i took 20 (one for each node)
- Total paths discover for 20 iterations is 4110
- So to find the average solution length I will divide for 400 because for each 20 iterations the algorithm internally loops 20 times:
- Average length = 4310/ 400 = 11
- So on average my Disikstra algorithm need to travel 11 nodes to find the distance between two nodes

A* Algorithm

It is a handy algorithm that is often used for map traversal to find the shortest path to be taken. A* was initially designed as a graph traversal problem, to help build a robot that can find its own course. It still remains a widely popular algorithm for graph traversal.

It searches for shorter paths first, thus making it an optimal and complete algorithm. An optimal algorithm will find the least cost outcome for a problem, while a complete algorithm finds all the possible outcomes of a problem

- uses heuristic to find the optimum path hear is how the algorithm works:
  1. Start from some initial node N

  2. Consider The neighbors of the node N with the lowest F score

  3. Add the other nodes to open list if they are not in there

  4. Cheek if neighbor is not visited before and is not destination node

  5. Repeat step 2 until the optimum path is found

Code analysis

- i found the shortest path from each node to the other all nodes so i iterate 400 times
- I run timit for 100,000 times to get the optimum average
- Total time i found for 100,000 iterations is 19.5 milliseconds
- so on average 400 iterations take 19.5 /100,000 = 1.95 milliseconds to run
- the average time for traveling from one node to another is therefore:
- Average time is 4.87

- so on average my A* algorithm takes 4.87 microsecond to travel from one node to the other

  Solution length analysis
- Number of iterations i took 400
- -Total paths discover for 20 iterations is 3534
- So to find the average solution length I will divide for 400 because for each 20 iterations the algorithm internally loops 20 times:
  Avg.length = 3534/ 400
  Avg.length = 9

- So on average my A* algorithm need to travel 9 nodes to find the distance between two nodes